1. **Write a java program to print the elements in Array?**

```
public class Test {

    public static void main(String[] args) {
        int arr[] = {1,9,8,3,4,6};
        for(int i=0;i<arr.length;i++) {
            System.out.print(arr[i]); //198346
        }

    }

}
```

2. **Sort array in ascending order?**

```
public class Test {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 4, 3, 5 };
        for (int i = 0; i < numbers.length; i++) {
            for (int j = i + 1; j < numbers.length; j++) {
                if (numbers[i] > numbers[j]) {
                    int temp = numbers[i];
                    numbers[i] = numbers[j];
                    numbers[j] = temp;
                }
            }
        }
        System.out.println(Arrays.toString(numbers)); //[1, 2, 3, 4, 5]
    }
}
```

3. **Sort array in descending order?**

```
public class Test {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 4, 3, 5 };
        for (int i = 0; i < numbers.length; i++) {
            for (int j = i + 1; j < numbers.length; j++) {
                if (numbers[i] < numbers[j]) {
                    int temp = numbers[i];
                    numbers[i] = numbers[j];
                    numbers[j] = temp;
                }
            }
        }
        System.out.println(Arrays.toString(numbers)); //[5, 4, 3, 2, 1]
    }
}
```

4. **Find Missing Number in Array?**

```java
public class Test {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 4, 5 };
        int n = numbers.length + 1; // Total numbers including the missing one
         System.out.println(n); //5
        int expectedSum = n * (n + 1) / 2; // Sum of first n natural numbers
        System.out.println(expectedSum); //15
        int actualSum = 0;
        for (int num : numbers) {
            actualSum += num;
        }
        System.out.println(expectedSum - actualSum); // 3
    }
}
```

5. **Find First Highest Number in Array?**

```java
public class Test {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 4, 5 };
        int max = numbers[0]; // Assume first element is the maximum
        for (int i = 1; i < numbers.length; i++) {
            if (numbers[i] > max) {
                max = numbers[i];
            }
        }

        System.out.println("The highest number is: " + max); //The highest number is: 5
    }
}
```

6. **Find Second Highest number in array?**

```java
int[] another = { 9,8,4,7,6 };
int highest = 0;
int secondHighest = 0;
for(int i =0; i<another.length;i++) {
    if(another[i]>highest) {
        secondHighest = highest;
        highest = another[i];
    } else if(another[i]>secondHighest) {
        secondHighest = another[i];
    }
}
System.out.println("Second Highest Number is " + secondHighest); //Second Highest Number is 8
```

7. **Java program for LongestSentenseWithOutRepeatingWord?**

```java
import java.util.Arrays;

public class LongestSentenseWithOutRepeatingWord {

    public static void main(String[] args) {
        String sentence = "the quick brown fox jumps over the lazy dog quick brownabc ";
        System.out.println(longestUniqueWords(sentence)); //brown fox jumps over the lazy dog quick brownabc
    }

    private static String longestUniqueWords(String sentence) {
        if (sentence == null || sentence.isEmpty()) {
            return null;
        }

        String words[] = sentence.split("\\s+");
        Set<String> wordsSet = new HashSet<>();
        int left = 0, maxStart = 0, maxLength = 0;

        for (int right = 0; right < words.length; right++) {
            while (wordsSet.contains(words[right])) {
                wordsSet.remove(words[left]);
                left++;
            }
            wordsSet.add(words[right]);

            if (right - left + 1 > maxLength) {
                maxLength = right - left + 1;
                maxStart = left;
            }

        }

        System.out.println(maxStart + maxLength);

        StringBuilder result = buildResult(words, maxStart, maxLength);

        System.out.println(result); //brown fox jumps over the lazy dog quick brownabc
        //Easy Approach
        return String.join(" ", Arrays.copyOfRange(words, maxStart, maxStart + maxLength));

    }

    private static StringBuilder buildResult(String[] words, int maxStart, int maxLength) {
        // Build result from maxStart to maxStart + maxLength
        StringBuilder result = new StringBuilder();
        for (int i = maxStart; i < maxStart + maxLength; i++) {
            result.append(words[i]);
            if (i < maxStart + maxLength - 1) {
                result.append(" ");
            }
        }
        return result;
    }
}
```

8. Java program for LongestSubStringWithOutRepeatingCharacter?

```java
public class LongestSubStringWithOutRepeatingCharacter {

    public static void main(String[] args) {
        String input = "abcabcbbabcde";
        String result = longestSubstring(input);
        System.out.println("Longest substring without repeating characters:");
        System.out.println(result); //Longest substring without repeating characters: abcde

    }

    private static String longestSubstring(String input) {
        int left =0,start=0,maxLength=0;
        Set<Character> charSet = new HashSet<>();

        for(int right=0;right<input.length();right++) {
            while(charSet.contains(input.charAt(right))) {
                charSet.remove(input.charAt(left));
                left ++;
            }
            charSet.add(input.charAt(right));

            if(right-left+1>maxLength) {
                maxLength = right-left+1;
                start = left;
            }
        }

        return input.substring(start,start+maxLength);
    }

}
```

9. **Java program for rate limiter?**

```java
import java.util.concurrent.atomic.AtomicInteger;

public class SimpleRateLimiter {
    private  int maxTokens;
    private long refillIntervalMillis;
    private long lastRefillTimestamp;
    private AtomicInteger availableTokens;

    public SimpleRateLimiter(int maxTokens, long refillIntervalMillis) {
        this.maxTokens = maxTokens;
        this.refillIntervalMillis = refillIntervalMillis;
        this.availableTokens = new AtomicInteger(maxTokens);
        this.lastRefillTimestamp = System.currentTimeMillis();
    }

    public synchronized boolean allowRequest() {
        refillTokens();

        if (availableTokens.get() > 0) {
            availableTokens.decrementAndGet();
            return true;
        }
        return false;
    }

    private void refillTokens() {
        long now = System.currentTimeMillis();
        long elapsed = now - lastRefillTimestamp;

        if (elapsed >= refillIntervalMillis) {
            int tokensToAdd = (int) (elapsed / refillIntervalMillis);
            int newTokenCount = Math.min(maxTokens, availableTokens.get() + tokensToAdd);
            availableTokens.set(newTokenCount);
            lastRefillTimestamp = now;
        }
    }

    public static void main(String[] args) throws InterruptedException {
        SimpleRateLimiter limiter = new SimpleRateLimiter(5, 1000); // 5 requests per second

        for (int i = 0; i < 10; i++) {
            if (limiter.allowRequest()) {
                System.out.println("Request " + i + " allowed");
            } else {
                System.out.println("Request " + i + " denied");
            }
            Thread.sleep(200); // simulate time between requests
        }
    }
}
```

Result:

```
Request 0 allowed
Request 1 allowed
Request 2 allowed
Request 3 allowed
Request 4 allowed
Request 5 allowed
Request 6 denied
Request 7 denied
Request 8 denied
Request 9 denied
```

10. Java program for TopNRepeatingWords?

```
public static void main(String[] args) {
    String[] arr = {"apple", "banana", "apple", "orange", "banana", "apple"};

    //Java 8 Solution
    System.out.println(Arrays.stream(arr).sorted().collect(Collectors.groupingBy(Function.identity(),LinkedHashMap::new,Collectors.counting()))
            .entrySet().stream()
            .filter(entry -> entry.getValue()>1).map(entry -> entry.getKey()).limit(2).collect(Collectors.toList())); //[apple, banana]

    //Regular Approach
    List<String> wordsList = Arrays.asList(arr);
    int top =2;
    wordsList = getTopNWords(wordsList,top);
    System.out.println(wordsList); //[apple, banana]
}
    private static List<String> getTopNWords(List<String> wordsList, int top) {
        if(null == wordsList || top <= 0) {
            return Collections.emptyList();
        }

        Map<String,Integer> frequencyMap = new HashMap<>();
        for(String word: wordsList) {
            frequencyMap.put(word, frequencyMap.getOrDefault(word, 0)+1);
        }

        List<Map.Entry<String, Integer>> sortedEntries = new ArrayList<>(frequencyMap.entrySet());
        sortedEntries.sort((k,v) -> {
            int frequencyCompare = v.getValue().compareTo(k.getValue());
            return frequencyCompare==0?frequencyCompare:k.getKey().compareTo(v.getKey());
        });

        List<String> result = new ArrayList<>();
        for (int i = 0; i < Math.min(top, sortedEntries.size()); i++) {
            result.add(sortedEntries.get(i).getKey());
        }

        return result;
    }

}
```

11. Java program for WatchCountService?

```
public class WatchCountService {
     // Thread-safe map to store videoId -> watch count
    private final ConcurrentHashMap<String, AtomicInteger> watchCounts = new ConcurrentHashMap<>();

    /**
     * Increments the watch count for a given video ID.
     */
    public void incrementWatchCount(String videoId) {
        watchCounts.computeIfAbsent(videoId, k -> new AtomicInteger(0)).incrementAndGet();
    }

    /**
     * Returns the watch count for a given video ID.
     */
    public int getWatchCount(String videoId) {
        return watchCounts.getOrDefault(videoId, new AtomicInteger(0)).get();
    }

    /**
     * Returns the entire map of video watch counts.
     */
    public ConcurrentHashMap<String, AtomicInteger> getAllWatchCounts() {
        return watchCounts;
    }
}
```

```java
public class WatchCountTest {

    public static void main(String[] args) {
        WatchCountService service = new WatchCountService();

        service.incrementWatchCount("video1");
        service.incrementWatchCount("video1");
        service.incrementWatchCount("video2");

        System.out.println("video1 views: " + service.getWatchCount("video1")); // video1 views: 2
        System.out.println("video2 views: " + service.getWatchCount("video2")); // video2 views: 1
        System.out.println("video2 views: " + service.getAllWatchCounts()); //video2 views: {video1=2, video2=1}

    }

}
```

12. Java algorithm to merge two unsorted arrays and sorted it in ascending order?

```java
public class MergeAndSort {
    public static void main(String[] args) {
        int[] arr1 = { 5, 2, 8, 4 };
        int[] arr2 = { 7, 1, 3 };

        int[] result = mergeAndSort(arr1, arr2);

        System.out.print("Merged & Sorted Array: ");
        for (int num : result) {
            System.out.print(num + " ");
        } //Merged & Sorted Array: 1 2 3 4 5 7 8
    }

    public static int[] mergeAndSort(int[] a, int[] b) {
        int n = a.length;
        int m = b.length;

        // Step 1: Merge into one array
        int[] merged = new int[n + m];
        for (int i = 0; i < n; i++) {
            merged[i] = a[i];
        }
        for (int j = 0; j < m; j++) {
            merged[n + j] = b[j];
        }

        // Step 2: Simple Bubble Sort (can replace with QuickSort or MergeSort)
        for (int i = 0; i < merged.length - 1; i++) {
            for (int j = 0; j < merged.length - i - 1; j++) {
                if (merged[j] > merged[j + 1]) {
                    int temp = merged[j];
                    merged[j] = merged[j + 1];
                    merged[j + 1] = temp;
                }
            }
        }

        return merged;
    }
}
```

13. **Java algorithm to merge two unsorted arrays with duplicates and sorted it in ascending order?**

```java
public class MergeSortRemoveDuplicates {

    public static void main(String[] args) {
        int[] arr1 = { 4, 5, 2, 8, 5 };
        int[] arr2 = { 7, 1, 2, 3, 1 };

        int[] result = mergeSortAndRemoveDuplicates(arr1, arr2);

        System.out.print("Merged, Unique & Sorted: ");
        for (int num : result) {
            System.out.print(num + " ");
        } //Merged, Unique & Sorted: 1 2 3 4 5 7 8

    }

    public static int[] mergeSortAndRemoveDuplicates(int[] a, int[] b) {
        // Step 1: Merge
        int[] merged = new int[a.length + b.length];
        for (int i = 0; i < a.length; i++)
            merged[i] = a[i];
        for (int i = 0; i < b.length; i++)
            merged[a.length + i] = b[i];

        // Step 2: Sort (Bubble Sort for simplicity)
        for (int i = 0; i < merged.length - 1; i++) {
            for (int j = 0; j < merged.length - 1 - i; j++) {
                if (merged[j] > merged[j + 1]) {
                    int temp = merged[j];
                    merged[j] = merged[j + 1];
                    merged[j + 1] = temp;
                }
            }
        }

        // Step 3: Remove duplicates
        int[] temp = new int[merged.length];
        int uniqueCount = 0;
        temp[uniqueCount++] = merged[0];

        for (int i = 1; i < merged.length; i++) {
            if (merged[i] != merged[i - 1]) {
                temp[uniqueCount++] = merged[i];
            }
        }

        // Step 4: Copy to final array
        int[] result = new int[uniqueCount];
        for (int i = 0; i < uniqueCount; i++) {
            result[i] = temp[i];
        }

        return result;
    }
}
```